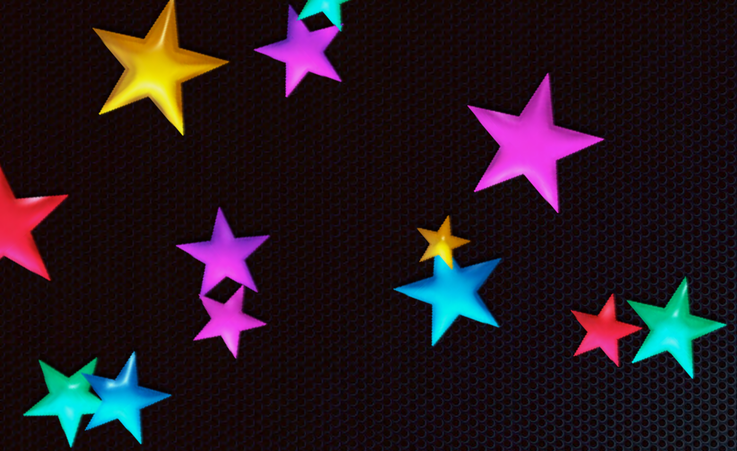


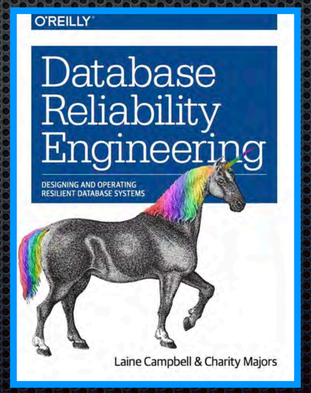
✨Observability✨Driven Development

@mipsytipsy
honeycomb

@mipsytipsy

engineer/cofounder/CTO

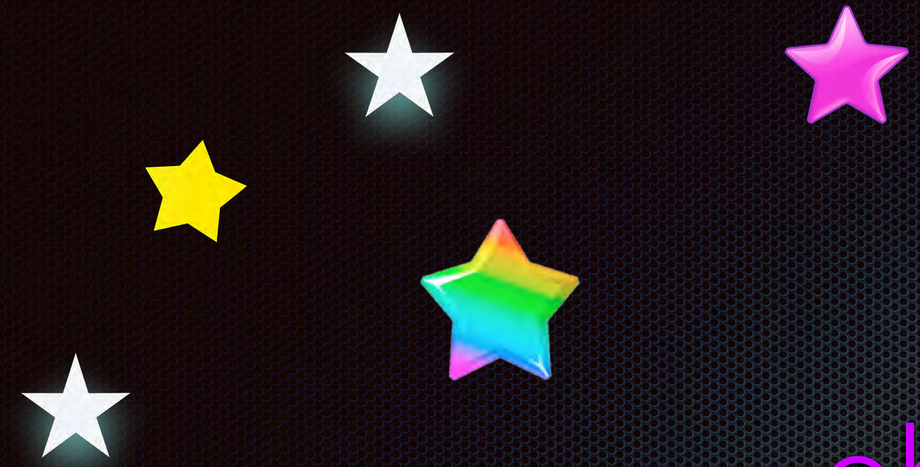
https://charity.wtf



honeycomb

Test-driven development (**TDD**) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the code is improved so that the tests pass

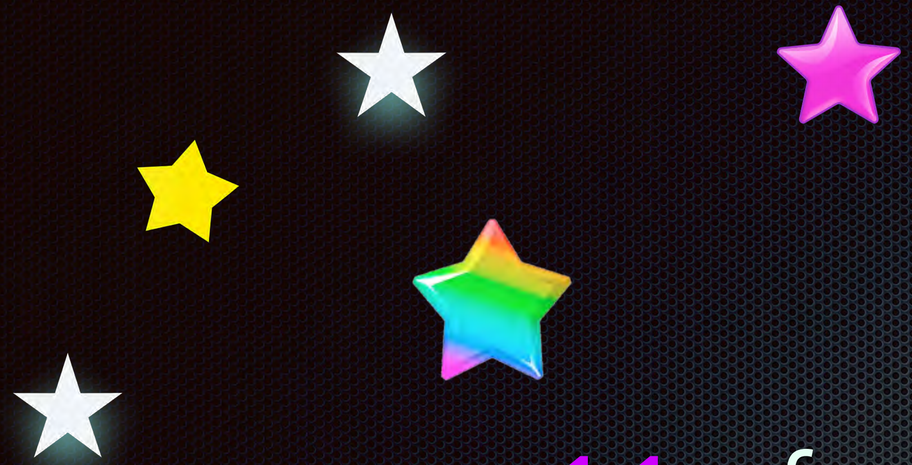
T.D.D. + Production == O.D.D.



observability(n):

"In control theory, observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. The observability** and controllability of a system are mathematical duals." – wikipedia

***observability is not monitoring, though both are forms of telemetry.*



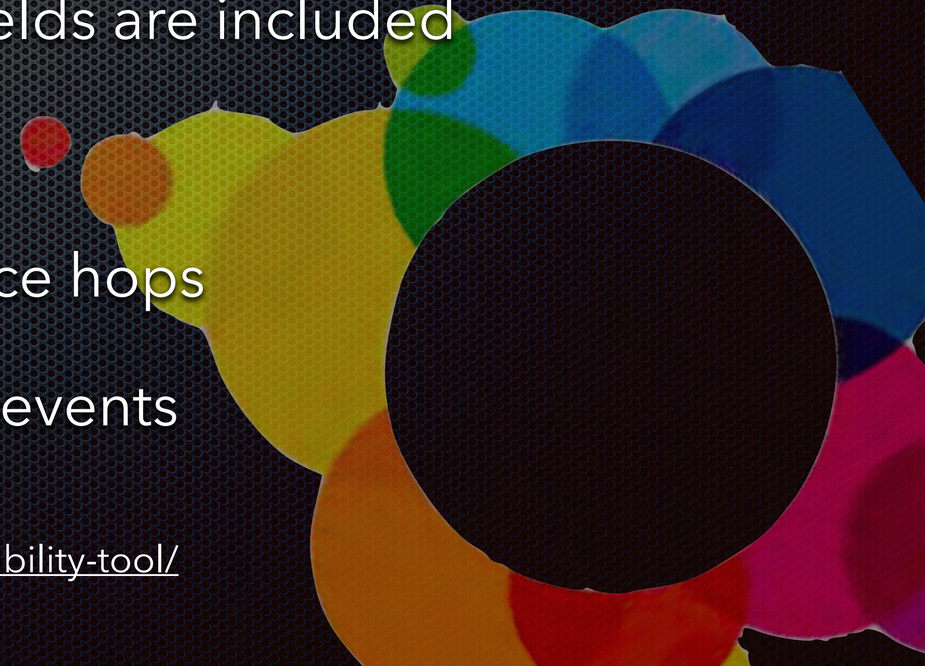
o11y for software engineers:

Can you understand what's happening inside your systems, just by asking questions from the outside? Can you figure out what transpired and identify any system state?

Can you answer any arbitrary new question …
without shipping new code?

The Bar: It's not observability unless it meets these reqs.

- High cardinality. High dimensionality
- Composed of arbitrarily-wide structured events (!metrics,! unstructured logs)
- Exploratory, open-ended investigation instead of dashboards
- Can visualize in waterfall trace by time if span_id fields are included
- No indexes, schemas, or predefined structure
- Bundles the full context of the request across service hops
- Aggregates only at compute/read time across raw events

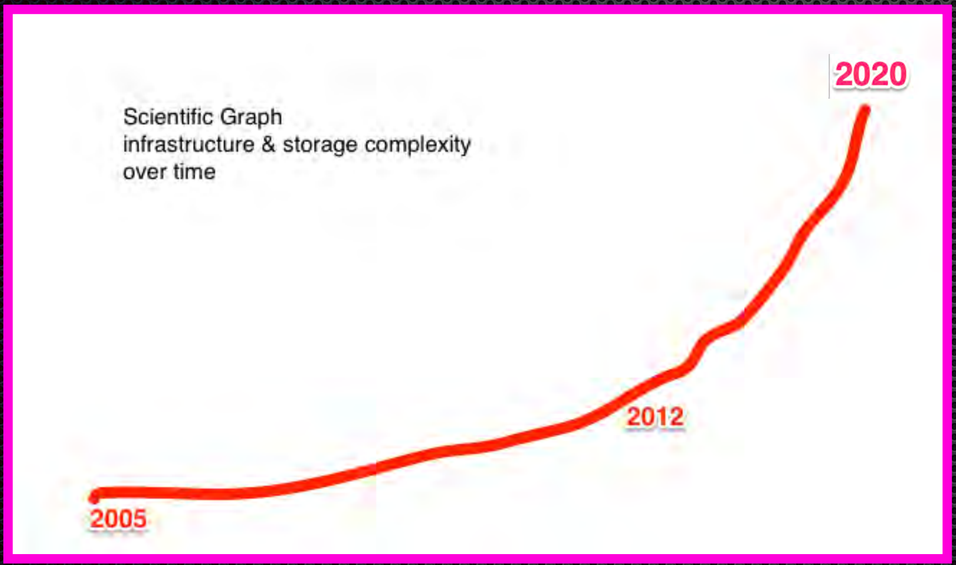


For more – read https://www.honeycomb.io/blog/so-you-want-to-build-an-observability-tool/

Why now?

Complexity is soaring;

the ratio of unknown-unknowns to known-unknowns has flipped

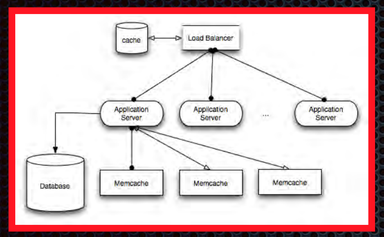


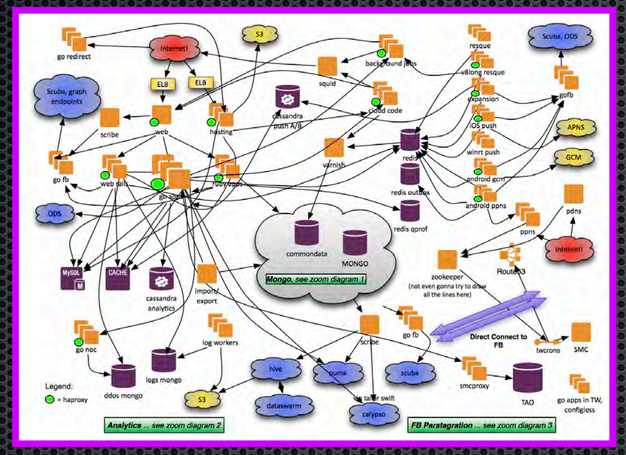
- Ephemeral and dynamic
- Far-flung and loosely coupled
- Partitioned, sharded
- Distributed and replicated
- Containers, schedulers
- Service registries
- Polyglot persistence strategies
- Autoscaled, multiple failover
- Emergent behaviors
- … etc

With a LAMP stack, you could lean on playbooks, guesses, pattern-matching and monitoring tools.

Now we have to instrument for observability.

or we are **screwed**





2003

2013

known-unknowns -> unknown-unknowns

Complexity is exploding everywhere,
but our tools were designed for a predictable world

Observability is the first step to high-performing teams because most teams are flying in the dark and don't even know it, and everything gets so much easier once you can SEE.WHERE.YOU.ARE.GOING.

They are using logs (where you have to know what you're looking for) or metrics (pre-aggregated and don't support high cardinality, so you can't ask any detailed question or iterate/drill down on a question).

Observability enables you to inspect cause and effect at a granular level – at the level of functions, endpoints and requests. This is a prerequisite for software engineers to own their code in production.

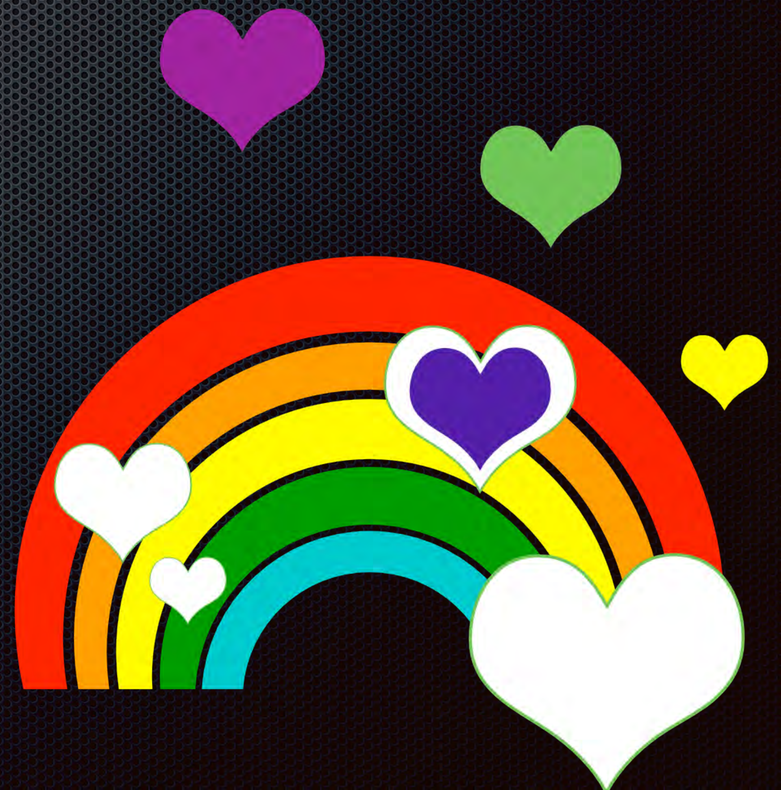
Without observability, your team must resort to guessing, pattern-matching and arguments from authority, and you will struggle to connect simple feedback loops in a timely manner.

It's like putting your glasses on before you drive off down the highway.

You have an observable system
when your team can quickly and reliably diagnose
any new behavior with no prior knowledge.

observability begins with
rich instrumentation, putting you in
constant conversation with your code

well-understood systems require
minimal time spent firefighting



never accept a pull request unless you can answer,

"how will I know when this breaks?" via your instrumentation

deploy one mergeset at a time. watch your code roll out,

then look thru the lens of your instrumentation and ask:

"working as intended? anything else look weird?"

"O.D.D."

and always wrap code in feature flags.

tools+processes

Use your tools and processes to
Practice Observability-Driven Development (ODD)
improve your tools and processes.

“if you change the tools people use, you can change how they behave and even who they are.”

engineer merges diff. hours pass, multiple other engineers merge too

someone triggers a deploy with a few days worth of merges

the deploy fails, takes down the site, and pages on call

who manually rolls back, then begins git bisecting

this eats up her day and multiple other engineers

everybody bitches about how on call sucks

insidious loop

50+ engineer-hours to ship this change

it doesn’t have to be that bad.

engineer merges diff, which kicks off an automatic CI/CD and deploy

deploy fails; notifies the engineer who merged, reverts to safety

who swiftly spots the error via his instrumentation

then adds tests & instrumentation to better detect it

and promptly commits a fix

virtuous loop:

eng time to ship this change: **10 min**

In order to spend more of your time on productive activities, instrument, observe, and iterate on the tools and processes that gather, validate and ship your collective output as a team.

Join teams that honor and value this work and are committed to consistently improving how they operate – not just shipping features.

Look for teams that are humble and relentlessly focused on investing in their core business differentiators.

look for ways to save time; ship smaller changesets more often

instrument, observe, measure before you act

connect output directly to the actor with context

shorten intervals between action and effect

instrument vigorously, boost negative signals

iterate and optimize

decouple deploys and releases

"I don't have time to invest in observability right now. Maybe later”

Ceej is sheltering under cats @ceejbot · Sep 4, 2019

We're making targeted fixes to what had been mysterious horrors in the backend and in the clients that call those endpoints.

This buys us time to do everything else we need to do.

3 4 20

This buys us time to do everything else we need to do.

I have no idea how anybody can *afford* to de-prioritize observability. Certainly we couldn't afford not to.

Information is power.

End of rant. Carry on.

5:48 PM · Sep 4, 2019 · Tweetbot for Mac

9 Retweets **49** Likes

You can't afford *not* to.

Charity Majors
@mipsytipsy